Элементы технического зрения ПРТС

LU – РАЗЛОЖЕНИЕ. БИБЛИОТЕКА EIGEN.

СЛАУ – что есть кроме Гаусса (и зачем это нужно)

- Рассмотрим метод Гаусса с более общих позиций

$$A \cdot x = b$$

$$A = \begin{pmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & & a_{2n} \\ \vdots & & \ddots & \vdots \\ a_{n1} & a_{n2} & \cdots & a_{nn} \end{pmatrix}, b = \begin{pmatrix} b_1 \\ b_2 \\ \vdots \\ b_n \end{pmatrix}$$

- Попробуем рассмотреть операцию исключения элементов первого столбца:

$$A^{(1)} = \begin{pmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ 0 & a_{22} & & a_{2n} \\ \vdots & & \ddots & \vdots \\ 0 & a_{n2} & \cdots & a_{nn} \end{pmatrix}$$

- Для этого мы вычисляли коэффициенты $\mu_{21}, \mu_{31}, \dots, \mu_{n1}$

СЛАУ – что есть кроме Гаусса (и зачем это нужно)

- Давайте введём матрицу

$$M_1 = \begin{pmatrix} 1 & 0 & 0 & \cdots & 0 \\ -\mu_{21} & 1 & 0 & \cdots & 0 \\ -\mu_{31} & 0 & 1 & \cdots & \\ \vdots & & & \ddots & \vdots \\ -\mu_{n1} & 0 & 0 & \cdots & 1 \end{pmatrix}$$

- Можно заметить, что $A^{(1)} = M_1 \cdot A$
- Аналогично

$$M_2 = \begin{pmatrix} 1 & 0 & 0 & \cdots & 0 \\ 0 & 1 & 0 & \cdots & 0 \\ 0 & -\mu_{32} & 1 & \cdots & \\ & \vdots & & \ddots & \vdots \\ 0 & -\mu_{n2} & 0 & \cdots & 1 \end{pmatrix}$$

LU-разложение

- В результате получим $A^{(m-1)} \cdot x = b^{(m-1)}$, где

$$A^{(m-1)} = M_{m-1}M_{m-2} \dots M_2M_1A$$
$$b^{(m-1)} = M_{m-1}M_{m-2} \dots M_2M_1b$$

- Можно выразить и в обратном направлении:

$$A = M_1^I M_2^I \dots M_{m-2}^I M_{m-1}^I A^{(m-1)}$$

- при этом

$$M_1^I = \begin{pmatrix} 1 & 0 & 0 & \cdots & 0 \\ \mu_{21} & 1 & 0 & \cdots & 0 \\ \mu_{31} & 0 & 1 & \cdots & \\ \vdots & & & \ddots & \vdots \\ \mu_{n1} & 0 & 0 & \cdots & 1 \end{pmatrix}$$

и т.д.

LU-разложение

- Перемножим матрицы:

$$M_1^I M_2^I \dots M_{m-2}^I M_{m-1}^I = \begin{pmatrix} 1 & 0 & 0 & \cdots & 0 \\ \mu_{21} & 1 & 0 & \cdots & 0 \\ \mu_{31} & \mu_{32} & 1 & \cdots & \\ & \vdots & & \ddots & \vdots \\ \mu_{n1} & \mu_{n2} & \mu_{n3} & \cdots & 1 \end{pmatrix} = L, A^{(m-1)} = U$$

- И, в результате, $A = LU$

$$Ax = LUx = b$$

LU-разложение

- В результате метод Гаусса можно разделить на 3 части:

Вычисляем матрицы L и U

Вычисляем $b^{(m-1)}=M^{I}b$

Решаем Ux=$b^{(m-1)}$

LU-разложение

- На первый взгляд мы усложнили себе жизнь
- Но, на самом деле мы просто выделили преобразование $b^{(m-1)}$
- За счёт этого мы можем повторять этапы 2 и 3 для РАЗНЫХ правых частей
- На практике метод Гаусса всегда реализуется через LU-разложение

Метод Холецкого (квадратного корня)

- Частный случай – матрица А является симметричной ($A^T = A$) и положительно определённой ($(Ax, x) > 0$)
- Таких случаев, на самом деле, довольно много
- Можно искать специальное представление

$$A = L \cdot L^T$$

- Метод примерно вдвое быстрее метода Гаусса, и гарантировано точнее за счёт меньшего накопления ошибки

Небольшое лирическое отступление - определитель

- А что, если нам всё таки хочется вычислить определитель?
- Считать его «в лоб» неэффективно: (n+1)! Действий
- Не в лоб: определитель треугольной матрицы равен произведению диагональных элементов
- Определитель не меняется при линейных преобразованиях строк
- Но меняет знак при их перестановке
- Следовательно,

$$\det A = (-1)^s \det U$$

- S – число перестановок строк (при неполном выборе главного элемента)

Eigen

- Пакет для матричных вычислений
- Матрицы, их преобразования, нормы и т.п.
- Методы решения СЛАУ

eigen.tuxfamily.org

Eigen

```
#include <iostream>
#define _USE_PREDEFINED_CONSTANT
#include <cmath>

#include <eigen3/Eigen/LU>
#include <eigen3/Eigen/Dense>
using std::cout;
using std::endl;

int main()
{
    Eigen::MatrixXd a(4,4);     //  Матрица бибиотеки Eigen
                                //  X - матрица произвольного размера
                                //  d -  элементами double
    Eigen::Matrix3f rotation;   //  Матрица 3x3 с float

    rotation << cos(M_PI_4), sin(M_PI_4), 0, -sin(M_PI_4), cos(M_PI_4), 0,\
            0, 0, 1;
    cout << "Rotation matrix\n" << rotation << endl;

    rotation(1,1) = cos(M_PI/6);
    cout << "Rotation matrix after modification\n" << rotation << endl;
    cout << "Now let's try access to matrix element in cycle" << endl;
    for (int i=0; i<rotation.rows(); i++)
    {
        for (int j=0; j<rotation.cols(); j++)
        {
            cout << ++rotation(i,j) << "\t";
        }
        cout << endl;
    }



    return 0;
}
```

Eigen

```
Rotation matrix
 0.707107  0.707107         0
-0.707107  0.707107         0
        0         0         1
Rotation matrix after modification
 0.707107  0.707107         0
-0.707107  0.866025         0
        0         0         1
Now let's try access to matrix element in cycle
1.70711 1.70711 1
0.292893        1.86603 1
1       1       2
Press <RETURN> to close this window...
```

Eigen - LU

```
#include <iostream>
#define _USE_PREDEFINED_CONSTANT
#include <cmath>

#include <eigen3/Eigen/LU>
#include <eigen3/Eigen/Dense>
using std::cout;
using std::endl;

int main()
{
    Eigen::MatrixXd a(4,4);     //  Матрица бибиотеки Eigen

    a << 10, 6, 2, 0, 5, 1, -2, 4, 3 ,5, 1, -1, 0, 6, -2, 2;
    cout << "Matrix A\n" << a << endl;
    Eigen::VectorXd b(4), x(4); //  Вектор - матрица с одним столбцом
    b << 25, 14, 10, 8;
    cout << "Vector b:\n" <<b << endl;

    x = a.lu().solve(b);
    cout << "Solution \n"<< x << endl;

    Eigen::VectorXd res = a*x-b;
    cout << "Residual vector\n" << std::fixed <<res << endl;
    cout << "Norm of residual vector " <<res.norm() << endl;


    return 0;
}
```

Eigen - LU

```
Matrix A
10  6  2  0
 5  1 -2  4
 3  5  1 -1
 0  6 -2  2
Vector b:
25
14
10
 8
Solution
   2
   1
-0.5
 0.5
Residual vector
0.000000
0.000000
0.000000
0.000000
Norm of residual vector 0.000000
Press <RETURN> to close this window...
```